A Survey on the Complexity of Flood-Filling Games

Michael R. Fellows[1], Frances A. Rosamond[1], Maise Dantas da Silva[2], and Uéverton S. Souza[2](✉)

[1] University of Bergen, Bergen, Norway
{michael.fellows,frances.rosamond}@uib.no
[2] Fluminense Federal University, Niterói, Brazil
maisedantas@id.uff.br, ueverton@ic.uff.br

Abstract. This survey is offered in honour of the special occasion of the birthday celebration of science and education pioneer Professor Juraj Hromkovič. In this survey, we review recent results on one-player flood-filling games on graphs, Flood-It and Free-Flood-It, in which the player aims to make the board monochromatic with a minimum number of *flooding moves*. As for many colored graph problems, flood-filling games have relevant interpretations in bioinformatics. The original versions of Flood-It and Free-Flood-It are played on $n \times m$ grids, but several studies were devoted to analyzing the complexity of these games when the "board" (the graph) belongs to other graph classes. A complete mapping of the complexity of flood-filling games on trees is presented, charting the consequences of single and aggregate parameterizations. The Flood-It problem on trees and the Restricted Shortest Common Supersequence (RSCS) problem are analogous. Flood-It remains NP-hard when played on 3-colored trees. A general framework for reducibility from Flood-It to Free-Flood-It is revisited. The complexity behavior of these games when played on various kinds of graphs is surveyed, such as Cartesian products of cycles and paths, circular grids, split graphs, co-comparability graphs, and AT-free graphs. We review a recent investigation of the parameterized complexity of Flood-It when the size of a minimum vertex cover is the structural parameter. Some educational aspects of the game are also reviewed. Happy Birthday, Juraj!

Keywords: Combinatorial games · Flood-filling games · Flood-It
Free-Flood-It · Graph algorithms · Parameterized complexity

1 Introduction

Flood-It is a one-player combinatorial game (also known as the Mad Virus Game), originally played on a colored board consisting of an $n \times m$ grid, where each tile/cell of the board has an initial color from a fixed color set. In the classic game, two tiles are *neighboring* tiles if they lie in the same row (resp. column) and in consecutive columns (resp. rows). A sequence C of tiles is a *path* when every

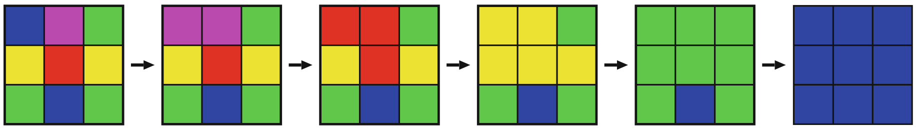

Fig. 1. An optimal sequence of moves to flood a 3×3 grid. (Color figure online)

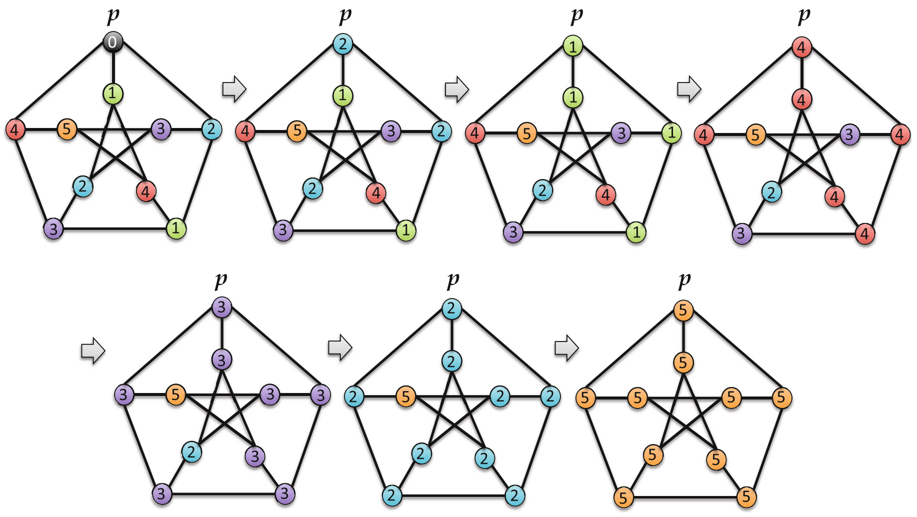

Fig. 2. An optimal sequence of moves to flood a 6-colored Petersen graph. (Color figure online)

pair of consecutive tiles in C is formed by neighboring tiles. A *monochromatic path* is a path in which all the tiles have the same color. Two tiles a and b are *m-connected* when there is a monochromatic path between them. In Flood-It, a move consists of assigning a new color c_i to the top left tile p (the *pivot*) and also to all the tiles m-connected to p immediately before the move. The objective of the game is to make the board monochromatic ("flood the board") with the minimum number of moves. Figure 1 shows a sequence of moves to flood a 3×3 grid colored with five colors.

A variation of Flood-It is Free-Flood-It, where the player can freely choose which tile will be the pivot of each move. In addition, these games can easily be generalized to be played on any graph with an initial coloring. Figure 2 shows a sequence of moves to flood a graph using a fixed pivot vertex p.

Many complexity issues regarding Flood-It and Free-Flood-It have recently been investigated. In [1], Arthur, Clifford, Jalsenius, Montanaro, and Sach show that Flood-It and Free-Flood-It are NP-hard on $n \times n$ grids colored with at least three colors. Meeks and Scott [28] prove that Free-Flood-It is solvable in polynomial time on $1 \times n$ grids, and also that Flood-It and Free-Flood-It remain NP-hard on $3 \times n$ grids colored with at least four colors.

Up to the authors' knowledge, the complexity of Flood-It on $3 \times n$ grids colored with three colors remains an open question. Clifford, Jalsenius, Montanaro, and Sach present in [7] a polynomial-time algorithm for Flood-It on $2 \times n$ grids. In [29], Meeks and Scott show that Free-Flood-It remains NP-hard on $2 \times n$ grids.

Fleischer and Woeginger [35] prove that Flood-It is NP-hard on trees. Analysing the complexity of Flood-It on non-grid graphs, Fleischer and Woeginger [14,35] also prove that Flood-It (also called the Honey-Bee-Solitaire problem) remains NP-hard even when restricted to split graphs, but that it is polynomial-time solvable on co-comparability graphs. When Flood-It is played on paths ($1 \times n$ grids) the problem is trivially solvable if the pivot has degree one. However, allowing the pivot be any vertex of the path, the problem is equivalent to the Shortest Common Supersequence problem (SCS) for two sequences [13,40], a very well-studied problem that does not have a known linear-time algorithm.

By similar arguments, it can be seen that Flood-It on cycles can be solved in polynomial time. In [28], Meeks and Scott show that Free-Flood-It on paths can be solved in $O(n^6)$ time. Thus, by the approach of removing the last vertex that will be flooded, we obtain an algorithm to solve Free-Flood-It on cycles in $O(n^7)$ time. In [25], it was shown that Free-Flood-It can be solved in polynomial time on 2-colored graphs.

In [30], it is shown the elegant fact that the minimum number of moves required to flood *any* given graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T. Meeks and Vu present some extremal properties of flood-filling games in [31]. Fukui et al. [15] show that if the number of colors is not bounded, Free-Flood-It is NP-complete even on caterpillars and proper interval graphs, but when the number of colors is a fixed constant, the game can be solved in XP time on interval graphs. Hon et al. [19] present a polynomial-time algorithm for Flood-It on AT-free graphs. In [41], Souza, Protti and Dantas da Silva describe polynomial-time algorithms to play Flood-It on C_n^2 (the second power of a cycle on n vertices), $2 \times n$ circular grids ($2 \times n$ grids where the first and last tiles in a same row are neighboring tiles). Souza et al. also show that Free-Flood-It is NP-hard on C_n^2 and $2 \times n$ circular grids. More recently, in [8,12] there was presented a parameterized complexity analysis of Flood-It, parameterizing with respect to the vertex cover number, and the neighborhood diversity of the input.

Flood-filling Games in Bioinformatics. Since the 90's, an increasing number of papers on biological applications have been dealt with as combinatorial problems. Vertex-colored graph problems have several applications in bioinformatics [9]. The Colored Interval Sandwich Problem has applications in DNA physical mapping [11,16] and in perfect phylogeny [27]; vertex-recoloring problems appear in protein-protein interaction networks and phylogenetic analysis [6,33]; the Graph Motif Problem [9] was introduced in the context of metabolic network analysis [24]; the Intervalizing Colored Graphs Problem [5] models DNA physical mapping [11]; and the Triangulating Colored Graph Problem [5] is polynomially equivalent to the Perfect Phylogeny Problem [17].

Flood-Filling games on colored graphs are also related to many problems in bioinformatics. As shown in [13,40], Flood-It played on trees is analogous to a restricted case of the Shortest Common Supersequence Problem [18]. Consequently, these games inherit from the Shortest Common Supersequence Problem many applications in bioinformatics, such as: microarray production [36], DNA sequence assembly [3], and a close relationship to multiple sequence alignment [38]. In addition, some disease spreading models, described in [2], work in a similar way to flood-filling games.

Additional Definitions and Notation. Neighboring tiles naturally correspond to neighboring vertices of a graph G representing the board; therefore, from now on, we use the term *vertex* instead of *tile*. A subgraph H of G is *adjacent* to a vertex $v \in V(G)$ if v has a neighbor in $V(H)$. A *flood move*, or just *move*, is a pair $m = (p, c)$ where p is the *pivot* of m (the vertex chosen to have its color changed by m), and c is the new color assigned to p; in this case, we also say that color c is *played in move* m. In Flood-It all moves have the same pivot. A subgraph H is said to be *flooded* when H becomes monochromatic. A vertex v is *flooded by a move* m if the color of v is played in m and v becomes m-connected to new vertices after playing m. We say that a move m *floods a vertex* v *by a vertex* w if v and w are neighbors and move m changes the color of w to flood v. A (free-)flooding is a sequence of moves in (Free-)Flood-It which floods G (the entire board). An optimal (free-)flooding is a flooding with a minimum number of moves. A move $m = (p, c)$ *is played on subgraph* H if $p \in V(H)$. An *island* is a vertex v colored with a color c such that no neighbor of v is colored with c. Let G_n be a graph with n vertices, the k-th power of G_n, denoted by G_n^k, is the graph formed by G_n plus edges between vertices at a distance at most k. Thus, P_n^k and C_n^k is the k-th power of a path P_n and a cycle C_n, respectively. A *circular grid* is an $n \times m$ grid with the additional property that the first and the last tiles in a same row are neighboring tiles. Finally, we denote by $\Pi \propto^f \Pi'$ a reduction from a problem Π to a problem Π' via a computable function f.

The formal definitions of the most studied flood-filling games are as follows.

Flood-It (decision version)
Instance: A colored graph G with a pivot vertex p, an integer λ.
Question: Is there a sequence of at most λ flood moves which makes the graph monochromatic, using p as the pivot in all moves?

Free-Flood-It (decision version)
Instance: A colored graph G, an integer λ.
Question: Is there a sequence of at most λ flood moves which makes the graph monochromatic?

Definition 1. *Let Π be a flood-filling game and let $S = \{s_1, \ldots, s_n\}$ be a subset of the aspects of Π (see below). $[S_1]$-$\Pi(S_2)$ is the family of parameterized problems where the aspects in $S_1 \subseteq S$ are fixed constants and the aspects in $S_2 \subseteq S$ are aggregate parameters.*

As an example, $[d]$-Flood-It(c) is the family of parameterized problems where d (maximum distance of the pivot) is a fixed constant and c (number of colors) is the parameter. We consider the following aspects of the problem:

- c = number of colors
- λ = number of moves
- d = maximum distance of the pivot
- o = maximum orbit
- t = number of leaves
- r = number of bad moves, $r = (\lambda - c)$

Given a vertex-colored graph G, the *orbit* of a color b in G, o_b, is the number of occurrences of b in G. We say the the maximum orbit of a vertex-colored graph G is the maximum orbit of a color used in G. A *good move* for a color c_a is a move that floods all non-flooded vertices with color c_a. A move that is not good is a *bad move*. As in Free-Flood-It there is no fixed pivot, for such a game the parameter d stands for the diameter of the graph.

2 Flood-Filling Games on Grids

In this section we summarize the main results on the complexity of flood-filling games on grids.

Lemma 1 (Clifford et al. [7]). *For $c \geq 4$, $[c]$-Flood-It and $[c]$-Free-Flood-It are NP-hard on grids. Further, for an unbounded number of colours c, there is no polynomial-time constant-factor approximation algorithm, unless $P = NP$.*

Lemma 2 (Clifford et al. [7]). *$[c = 3]$-Flood-It and $[c = 3]$-Free-Flood-It on grids are NP-hard.*

Theorem 1 (Clifford et al. [7]). *Flood-It is polynomial-time solvable on $2 \times n$ grids.*

Theorem 2 (Meeks and Scott [28]). *$[c = 4]$-Flood-It remains NP-hard when restricted to $3 \times n$ grids.*

Theorem 3 (Meeks and Scott [28]). *$[c = 4]$-Free-Flood-It remains NP-hard when restricted to $3 \times n$ grids.*

Theorem 4 (Meeks and Scott [28]). *Free-Flood-It can be solved in polynomial time on $1 \times n$ grids.*

Theorem 5 (Meeks and Scott [28], Lagoutte et al. [25]). *$[c = 2]$-Free-Flood-It can be solved in polynomial time on general graphs.*

Theorem 6 (Meeks and Scott [29]). *Free-Flood-It remains NP-hard when restricted to $2 \times n$ grids.*

Theorem 7 (Meeks and Scott [29]). *Free-Flood-It(c) can be solved in $O(2^c \times n^{10})$ time on $2 \times n$ grids.*

In [7], Clifford et al. also provided a discussion about approximating the number of moves, and grids where each tile is coloured uniformly at random. Up to the authors' knowledge, the complexity of Flood-It and Free-Flood-It on $k \times n$ $(k \geq 3)$ grids colored with three colors remains as an open question.

Tables 1 and 2 summarize the results of this section.

Table 1. Complexity of Flood-It

#colors\ grid size	$1 \times n$	$2 \times n$	$3 \times n$	$k \times n$	$n \times n$
2	P	P	P	P	P
3	P	P	open	open	NP-hard [7]
4	P	P	NP-hard [28]	NP-hard	NP-hard
unbounded	P	P [7]	NP-hard	NP-hard	NP-hard

Table 2. Complexity of Free-Flood-It

#colors\ grid size	$1 \times n$	$2 \times n$	$3 \times n$	$k \times n$	$n \times n$
2	P	P	P	P	P [25,28]
3	P	P	open	open	NP-hard [7]
4	P	P [29]	NP-hard [28]	NP-hard	NP-hard
unbounded	P [28]	NP-hard [29]	NP-hard	NP-hard	NP-hard

3 Flood-Filling Games on Trees

Now, we revisit a multivariate investigation of the complexity of Flood-It and Free-Flood-It when played on trees presented in [13,40], where the authors analyze the complexity consequences of parameterizing flood-filling problems in various ways.

3.1 Flood-It on Trees

We start this section by remarking that Flood-It played on a tree is equivalent to Flood-It played on a rooted tree whose root is the pivot.

Theorem 8 (Fellows et al. [13], Souza et al. [40]). *$[d]$-Flood-It on trees remains NP-hard when $d = 2$.*

The proof of Theorem 8 uses a reduction from the Vertex Cover Problem. The authors show that there is a vertex cover of size k in a graph G if and only if there is a flooding with $n + k$ moves in the associated tree T. Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, they construct a tree T as follows:

- create a pivot root s with color c_s;
- for each edge $e_i = uv$ of G, add to T a subset of vertices $E_i = \{u'_i, v'_i, u''_i, v''_i\}$ such that u'_i, v'_i are children of s, v''_i is a child of u'_i, and u''_i is a child of v'_i;
- define a distinct color c_u for each $u \in V(G)$, and color all vertices of the form u'_i, u''_i (for all i) with the color c_u.

Figure 3 shows a graph G and its associated tree T.

Theorem 8 shows that the problem remains NP-hard even for a very restricted class of trees. Notice that Flood-It and Free-Flood-It are trivially solvable when T is a star.

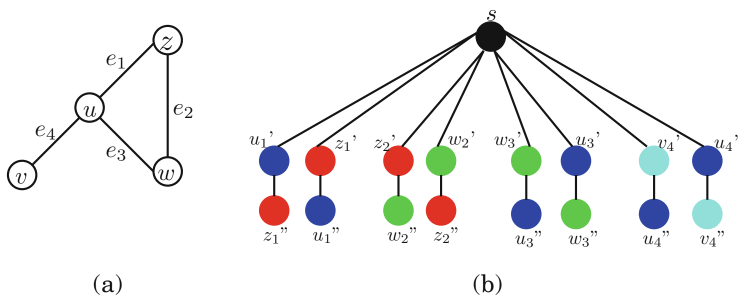

Fig. 3. (a) A graph G; (b) tree T obtained from G.

Corollary 1. *$[o, d]$-Flood-It on trees is NP-hard even when $o = 4$ and $d = 2$.*

Corollary 1 follows by restricting the reduction presented in Theorem 8 to cubic graphs. In addition, in [13,40] the authors also show the following result.

Theorem 9. *$[d]$-Flood-It(c) is in FPT and admits a polynomial kernelization.*

Analogous Problems

Next we present a very interesting observation provided in [13].

Definition 2. *Two optimization problems Π and Π' are said to be* analogous *if there exist linear-time reductions f, g such that:*

1. *$\Pi \propto^f \Pi'$ and $\Pi' \propto^g \Pi$;*
2. *every feasible solution s for an instance I of Π implies a feasible solution s' for $f(I)$ such that size(s) = size(s');*
3. *every feasible solution s' for an instance I' of Π' implies a feasible solution s for $g(I')$ such that size(s') = size(s).*

Next we have an equivalent definition for decision problems. Denote by $Y(\Pi)$ the set of all instances I of Π yielding a yes-answer for the question "$I \in Y(\Pi)$?".

Definition 3. *Two decision problems Π and Π' in NP are said to be* analogous *if there exist linear-time reductions f, g such that:*

1. *$\Pi \propto^f \Pi'$ and $\Pi' \propto^g \Pi$;*
2. *every easy checkable certificate $\mathcal{C}$ for the yes-answer of the question "$I \in Y(\Pi)$?" implies an easy checkable certificate $\mathcal{C}'$ for the yes-answer of the question "$f(I) \in Y(\Pi')$?" such that $size(\mathcal{C}) = size(\mathcal{C}')$;*
3. *every easy checkable certificate $\mathcal{C}'$ for the yes-answer of the question "$I' \in Y(\Pi')$?" implies an easy checkable certificate $\mathcal{C}$ for the yes-answer of the question "$g(I') \in Y(\Pi)$?" such that $size(\mathcal{C}') = size(\mathcal{C})$.*

Definition 4. *Let Π and Π' be analogous decision problems. The parameterized problems $\Pi(k_1, \ldots, k_t)$ and $\Pi'(k'_1, \ldots, k'_t)$ are said to be* p-analogous *if there exist FPT reductions f, g and a one-to-one correspondence $k_i \leftrightarrow k'_i$ such that:*

1. *$\Pi(k_1, \ldots, k_t) \propto^f \Pi'(k'_1, \ldots, k'_t)$ and $\Pi'(k'_1, \ldots, k'_t) \propto^g \Pi(k_1, \ldots, k_t)$;*
2. *every easy checkable certificate $\mathcal{C}$ for the yes-answer of the question "$I \in Y(\Pi(k_1, \ldots, k_t))$?" implies an easy checkable certificate $\mathcal{C}'$ for the yes-answer of the question "$f(I) \in Y(\Pi'(k'_1, \ldots, k'_t))$?" such that $k'_i = \varphi'_i(k_i)$ for some function φ'_i $(1 \leq i \leq t)$;*
3. *every easy checkable certificate $\mathcal{C}'$ for the yes-answer of the question "$I' \in Y(\Pi'(k'_1, \ldots, k'_t))$?" implies an easy checkable certificate $\mathcal{C}$ for the yes-answer of the question "$g(I') \in Y(\Pi(k_1, \ldots, k_t))$?" such that $k_i = \varphi_i(k'_i)$ for some function φ_i $(1 \leq i \leq t)$.*

Two straightforward consequences of the above definitions are: (a) if Π and Π' are analogous problems then Π is in P (is NP-hard) if and only if Π' is in P (is NP-hard); (b) if $\Pi(k_1, \ldots, k_\ell)$ and $\Pi'(k'_1, \ldots, k'_\ell)$ are p-analogous problems then $\Pi(k_1, \ldots, k_\ell)$ is in FPT (admits a polynomial kernel/is W[1]-hard) if and only if $\Pi'(k'_1, \ldots, k'_\ell)$ is in FPT (admits a polynomial kernel/is W[1]-hard).

Fleischer and Woeginger used a reduction from the Fixed Alphabet Shortest Common Supersequence Problem [35] to prove that Flood-It on trees is NP-hard even when the number of colors is fixed. In [13], the authors show that Flood-It on trees and Restricted Shortest Common Supersequence(RSCS) are analogous problems. RSCS is a variant of SCS - Shortest Common Supersequence [10].

Shortest Common Supersequence (SCS)
(decision version)
Instance: A set of strings $S = s_1, \ldots, s_\ell$ over an alphabet Σ, an integer Λ.
Question: Does there exist a string $s \in \Sigma$ of length at most Λ that is a supersequence of each string in S?

Restricted Shortest Common Supersequence (RSCS)
(decision version)
Instance: A set of ρ-strings $R = r_1, \ldots, r_\ell$ over an alphabet Σ, an integer Λ. (A ρ-string is a string with no identical consecutive symbols.)
Question: Does there exist a string $r \in \Sigma$ of length at most Λ that is a supersequence of each ρ-string in R?

Let SCS($|\Sigma_1|, \ell_1$) stand for the SCS problem parameterized by $|\Sigma_1|$ and ℓ_1 (ℓ_1 is the number of strings). The notation RSCS($|\Sigma_2|, \ell_2$) is used similarly.

Theorem 10 (Fellows et al. [13]). *SCS*($|\Sigma_1|, \ell_1$) *is FPT-reducible to RSCS* ($|\Sigma_2|, \ell_2$).

Theorem 11 (Fellows et al. [13]).

(a) Flood-It on trees and RSCS are analogous problems.
(b) Flood-It(c, t, λ) *on trees is p-analogous to RSCS*($|\Sigma|, \ell, \Lambda$).

By Theorem 11, results valid for RSCS can be inherited by Flood-It on trees:

Corollary 2 (Fellows et al. [13]). $[t]$*-Flood-It on trees is solvable in polynomial time.*

Corollary 3 (Fellows et al. [13]). *Flood-It*(t, c) *on trees is W[1]-hard.*

Phylogenetic Colored Trees

Flood-It played on trees can be applied to scheduling. Each color corresponds to an operation in the sequential process of manufacturing an object. In the input tree T, paths from the pivot to the leaves correspond to the manufacturing sequences for a number of different objects that share the same production line. A flooding to T then corresponds to a schedule of operations for the production line that allows all of the different objects to be manufactured. It may reasonably be the case that each object to be manufactured requires any given operation to be applied at most once.

Theorem 12 (Fellows et al. [13]). $[r]$*-Flood-It on general graphs can be solved in polynomial time.*

Definition 5. *A colored rooted tree is a pc-tree (phylogenetic colored tree) if no color occurs more than once in any path from the root to a leaf.*

Corollary 4 (Fellows et al. [13]). *Flood-It on trees remains NP-hard even when restricted to pc-trees with pivot root.*

Corollary 5 (Fellows et al. [13]). *Flood-It*(t) *on pc-trees with pivot root is W[1]-hard.*

Definition 6. *A pc-tree T is a cpc-tree (complete pc-tree) if each color occurs exactly once in any path from the root to a leaf.*

Cpc-trees are a special subclass of pc-trees. Many hard cases of Flood-It on pc-trees are easy to solve when restricted to cpc-trees; for example, while Flood-It on pc-trees remains NP-hard when d is the parameter, Flood-It on cpc-trees is trivially solved in FPT time.

As in biological applications the phylogenetic sequences are often complete, the complexity of flood-filling games for complete pc-trees is an interesting issue.

Theorem 13 (Fellows et al. [13]). *Flood-It on trees remains NP-hard even when restricted to cpc-trees with pivot root.*

In Corollary 1, it was shown that Flood-It remains NP-hard even when restricted to pc-trees with maximum orbit 4. Theorem 14 shows that Flood-It on cpc-trees can be solved in polynomial time when considering the maximum orbit as the parameter.

Theorem 14 (Fellows et al. [13]). $[o]$*-Flood-It on cpc-trees can be solved in polynomial time.*

Flood-It on 3-Colored Trees

Flood-It on 2-colored graphs is trivially solvable. Fleischer and Woeginger [35] proved that Flood-It remains NP-hard when restricted to 4-colored trees. Raiha and Ukkonen [37] proved that Shortest Common Supersequence over a binary alphabet is NP-complete, and Middendorf [32] proved that Shortest Common Supersequence over a binary alphabet remains NP-complete even if the given strings have the same length and each string contains exactly two ones.

In Middendorf's proof the instances of Shortest Common Supersequence do not have two consecutive ones; hence, without loss of generality, we can assume that the last character of each input string is '0' (since after each '1' there is a '0'). Using this fact, in [13] was proved the following theorem.

Theorem 15 (Fellows et al. [13]). $[c]$*-Flood-It on trees remains NP-hard when* $c = 3$.

3.2 Free-Flood-It on Trees

Theorem 16 (Fellows et al. [13]). $[r]$*-Free-Flood-It on general graphs can be solved in polynomial time.*

The proof of Theorem 16 is similar to the proof presented in Theorem 12 in [13].

Now we present a general framework for reducibility from Flood-It to Free-Flood-It provided in [13].

Definition 7. *Let* G *be a graph,* $v \in V(G)$*, and* ℓ *a positive integer. The graph* $\psi(G, v, \ell)$ *is constructed as follows:* (i) *create* ℓ *disjoint copies* $G_1, \ldots, G_\ell$ *of* G*;* (ii) *contract the copies* $v_1, v_2, \ldots, v_\ell$ *of* v *into a single vertex* v^*.

Definition 8. *Let* $\mathscr{F}$ *be a class of graphs. Then:*

$$\psi(\mathscr{F}) = \{G \mid G = \psi(G', v, \ell) \text{ for some triple } (G' \in \mathscr{F}, v \in V(G'), \ell > 0)\ \}.$$

Definition 9. *A class* $\mathscr{F}$ *of graphs is closed under operator* ψ *if* $\psi(\mathscr{F}) \subseteq \mathscr{F}$.

Examples of classes closed under ψ are chordal graphs and bipartite graphs.

Theorem 17 (Fellows et al. [13]). *Flood-It played on $\mathscr{F}$ is reducible in polynomial time to Free-Flood-It played on $\psi(\mathscr{F})$.*

Corollary 6 (Fellows et al. [13]). *Let $\mathscr{F}$ be a class of graphs closed under ψ. Then Flood-It played on $\mathscr{F}$ is reducible in polynomial time to Free-Flood-It played on $\mathscr{F}$.*

NP-hardness results valid for Flood-It can be inherited by Free-Flood-It:

Corollary 7 (Fellows et al. [13]). $[d]$*-Free-Flood-It on trees is NP-hard even when $d = 4$.*

Corollary 8 (Fellows et al. [13]). *Free-Flood-It on cpc-trees is NP-hard.*

Corollary 9 (Fellows et al. [13]). $[c]$*-Free-Flood-It on trees is NP-hard even when $c = 3$.*

The next theorem implies that Flood-It on pc-trees and Free-Flood-It on pc-trees are analogous, and parameterized versions of these problems are p-analogous.

Theorem 18. *In Free-Flood-It on pc-trees, there always exists an optimal free-flooding which is a flooding with pivot root.*

Corollary 10 (Fellows et al. [13]). *Free-Flood-It*(t) *on pc-trees is W[1]-hard.*

Corollary 11 (Fellows et al. [13]). *Free-Flood-It*(t, r) *on pc-trees with pivot root is in FPT.*

Table 3 summarizes the results presented in [13].

4 Flood-Filling Games on Other Classes of Graphs

In this section we consider the complexity of flood-filling games played on other classes of boards, such as split graphs, co-comparability graphs, powers of paths, powers of cycles and circular grids.

In [35], Flood-It was denoted by Honey-Bee-Solitaire.

Theorem 19 (Fleischer and Woeginger [35]). *Flood-It can be solved in polynomial time on co-comparability graphs.*

Theorem 20 (Fleischer and Woeginger [35]). *Flood-It on split graphs is NP-hard.*

Recently, Hon et al. [19] claim that Flood-It on AT-free graphs can be solved in polynomial time.

Theorem 21 (Hon et al. [19]). *Flood-It can be solved in polynomial time on AT-free graphs graphs.*

Table 3. Multivariate analysis of flood-filling games on trees.

Problem	Instance	Fixed constant	Parameters	Complexity
Flood-It	cpc-trees		–	NP-hard
	trees	$c \geq 3$	–	NP-hard
	pc-trees		c	FPT
	graphs		λ	FPT
	cpc-trees		d	FPT
	cpc-trees	o	–	Polynomial
	pc-trees		k	W[1]-hard
	graphs	r	–	Polynomial
	cpc-trees		r	FPT
	trees		c, d	FPT
	graphs		c, o	FPT
	trees		c, k	W[1]-hard
	trees		c, r	FPT
	pc-trees	$d \geq 2$, $o \geq 4$	–	NP-hard
	trees		d, k	FPT
	trees		o, k	W[1]-hard
	trees		k, r	FPT
Free-Flood-It	cpc-trees		–	NP-hard
	trees	$c \geq 3$	–	NP-hard
	trees	$d \geq 4$	–	NP-hard
	pc-trees		k	W[1]-hard
	cpc-trees		r	FPT
	trees		o, k	W[1]-hard
	pc-trees		k, r	FPT

The next results focus on the powers of some classes of graphs, and circular grids.

Corollary 12 (Souza et al. [41]). *Flood-It is solvable in polynomial time on $2 \times n$ circular grids.*

Lemma 3 (Souza et al. [41]). *Flood-It on C_n^2 is a particular case of Flood-It on circular grids.*

Figure 4 illustrates a C_n^2 as a particular instance of Flood-It on circular grids.

Corollary 13 (Souza et al. [41]). *Flood-It can be solved in polynomial time on C_n^2.*

Corollary 14 (Souza et al. [41]). *Flood-It can be solved in polynomial time on P_n^2.*

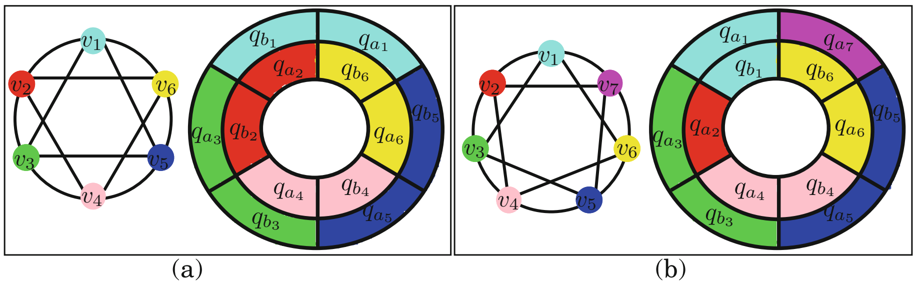

Fig. 4. (a) $2 \times n$ circular grid for even n; (b) $2 \times n$ circular grid for odd n.

In [31], Meeks and Vu present some upper bounds to the maximum number of moves that might be required to flood a arbitrary graph, which they show to be tight for particular classes of graphs. They also determine this maximum number of moves exactly when the underlying graph is a path, cycle, or a blow-up of a path or cycle (Table 4).

Table 4. Complexity of flood-filling games on particular graph classes

	Flood-It	Free-Flood-It
Caterpillars	P [35]	NP-hard [15]
P_n^2	P [41]	NP-hard [41]
Proper interval	P [35]	NP-hard [15]
Co-comparability	P [35]	NP-hard [15]
AT-free	P [19]	NP-hard [15]
C_n^2	P [41]	NP-hard [41]
$2 \times n$ circular grid	P [41]	NP-hard [41]
Split	NP-hard [35]	NP-hard [15]

Free-Flood-It

Theorem 22 (Fukui et al. [15]). *Free-Flood-It is NP-hard even on proper interval graphs, or even on caterpillars. These results still hold even if the maximum degree of the graphs is bounded by 3.*

Theorem 23 (Fukui et al. [15]). *Free-Flood-It on an interval graph can be solved in $O(4^c c^2 n^3)$ time.*

Notice that Theorem 23 shows that the problem is fixed-parameter tractable on interval graphs considering the number of colors as the parameter.

Flood-It on paths can be easily solved in $O(n^2)$ time by a dynamic programming [40], and as show in [41], the problem remains polynomially solvable when played on $2 \times n$ circular grids, C_n^2 and P_n^2. Although Free-Flood-It can be solved in polynomial time when played on paths and cycles, in [41], they show that Free-Flood-It is NP-hard when played on C_n^2, P_n^2 or circular grids.

Theorem 24 (Souza et al. [41]). *Free-Flood-It remains NP-hard on* C_n^2.

Corollary 15 (Souza et al. [41]). *Free-Flood-It remains NP-hard on* P_n^2.

Corollary 16 (Souza et al. [41]). *Free-Flood-It remains NP-hard on* $2 \times n$ *circular grids.*

5 The Size of a Minimum Vertex Cover as Parameter

From the parameterized complexity point of view, we consider the complexity of the Flood-It game played on graphs with bounded minimum vertex cover. We revisit the results presented in [8,12].

In the literature, there exist some results considering bounded values for different parameters of Flood-It. For instance, Flood-It is NP-hard on $n \times n$ grids colored with at least three *colors* [1], and it is also NP-hard on trees with *diameter* at most four [13,40]. In [13,40], the authors show some parameterized complexity results on Flood-It on trees, such as: Flood-It is W[1]-hard on trees when the number of leaves and the number of colors are parameters. On the other hand, it is easy to verify in $O^*(\lambda^\lambda)$ time whether Flood-It has a solution of size at most λ, and to obtain a kernel of size $O(c^d)$ where c and d are the parameters, the number of colors and the diameter of the graph, respectively. At this point, we review the parameterized complexity of Flood-It game considering the minimum vertex cover of the board (graph) as the parameter.

In [8,12], they present a parameterized complexity analysis of Flood-It with respect to the vertex cover number and with respect to the neighborhood diversity.

Theorem 25 (Fellows et al. [8,12]). *Flood-It is in FPT when parameterized by the vertex cover number* ($|vc|$).

Definition 10. *A graph* $G(V,E)$ *has neighborhood diversity* $nd(G) = t$ *if one can partition* V *into* t *sets* $V_1, \ldots, V_t$ *such that, for all* $v \in V$ *and every* $i \in \{1, \ldots, t\}$, *either* v *is adjacent to every vertex in* V_i *or it is adjacent to none of them. Note that each part* V_i *of* G *is either a clique or an independent set.*

The parameter neighborhood diversity is a natural generalization of the vertex cover number. In 2012, Lampis [26] showed that for every graph G we have $nd(G) \leq 2^{|vc|} + |vc|$, where $|vc|$ is the vertex cover number of G. The optimal neighborhood diversity decomposition of a graph G can be computed in $\mathcal{O}(n^3)$ time [26].

Corollary 17 (Fellows et al. [8,12]). *Flood-It is fixed-parameter tractable when parameterized by the neighborhood diversity.*

Theorem 26 (Fellows et al. [8,12]). *Flood-It admits polynomial kernelization when parameterized by the neighborhood diversity* (nd) *and the number of colors* (c).

Theorem 27 (Fellows et al. [8,12]). *Flood-It parameterized by the vertex cover number does not admit a polynomial kernel, unless coNP* $\subseteq$ *NP/poly, even restricted to bipartite or chordal graphs.*

Corollary 18 (Fellows et al. [8,12]). *Flood-It does not admit a polynomial kernel, unless coNP* $\subseteq$ *NP/poly, even when the vertex cover number and the maximum number of bad moves to be played are considered as an aggregate parameter.*

Based on the Exponential Time Hypothesis (ETH) and the Strong Exponential Time Hypothesis (SETH), in [12] they obtained the following bounds for Flood-It.

Theorem 28 (Fellows et al. [12]). *Flood-It cannot be solved in*

1. $2^{o(|vc|+i_c)}n^{\mathcal{O}(1)}$ *time, unless ETH fails, and*
2. $(2-\varepsilon)^{i_c}n^{\mathcal{O}(1)}$ *time, unless SETH fails,*

even when the input graph is either bipartite or chordal, and i_c *denotes the minimum number of colors of a maximum independent set of* G.

Theorem 28 provides a strong bound for bipartite and chordal graphs. The next result gives us bounds for very restricted subclasses of bipartite and chordal graphs.

Theorem 29 (Fellows et al. [12]). *Unless ETH fails, Flood-It cannot be solved in* $2^{o(|vc|+i_c)}n^{\mathcal{O}(1)}$ *time, even when the input graph* G *is a tree with height two or a split graph formed by a clique and a set of pendant vertices.*

In [12] the author also provided an exact algorithm for Flood-It.

Theorem 30 (Fellows et al. [12]). *Flood-It can be solved in* $\mathcal{O}(2^{\mathcal{O}(|vc|\, log(i_c\, |vc|))} n^{\mathcal{O}(1)})$ *time, where* $|vc|$ *is the vertex cover number and* i_c *is the minimum number of colors of a maximum independent set of* G.

In [30] they show that the minimum number of moves required to flood any given graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T. This result can be applied to give polynomial-time algorithms for flood-filling problems.

Corollary 19 (Meeks and Scott [30]). *Free-Flood-It is solvable in polynomial time on subdivisions of any fixed graph* H.

6 Final Considerations

We revisited recent results (see [1,7,8,12–15,19,25,28–31,35,40,41]) on flood-filling games. Many complexity issues on Flood-It and Free-Flood-It have recently been investigated, and these games have presented interesting behavior when played on non-grid graphs. We also briefly presented a multivariate investigation of the complexity of Flood-It and Free-Flood-It when played on trees provided in [13,39,40]. During our analysis we remember that Flood-It on trees is analogous to Restricted Shortest Common Supersequence, and Flood-It remains NP-hard on 3-colored trees. We also revisited a general framework for reducibility from Flood-It to Free-Flood-It. From the parameterized complexity point of view, we revisited recent results which show that: Flood-It game played on graphs with bounded minimum vertex cover is fixed-parameter tractable, and admits polynomial kernelization when besides the minimum vertex cover, the number of colors is also part of the parameter.

6.1 Open Problems

We present some open problems on flood-filling games:

– What are the complexities of $[c = 3]$-Flood-It and $[c = 3]$-Free-Flood-It on $k \times n$ grids, in the case that $k \geq 3$ is a fixed integer?
– Does Flood-It(r) remain fixed-parameter tractable on general pc-trees?

Motivated by the general framework for reducibility from Flood-It to Free-Flood-It presented in [13], the following question arises:

– Is there any graph class for which Free-Flood-It can be solved in polynomial time, but Flood-It is NP-hard?

In addition, it is interesting to identify new graph classes for which Flood-It can be non-trivially solvable in polynomial time. It is also interesting to identify other single parameters for which Flood-It is fixed-parameter tractable.

We could think of games as a way of defining graph parameters. The paper [34], gives a game-defined look at graph structure. From this perspective, they show that it follows that branch-width is polynomially computable for planar graphs. Maybe a parameter such as "flood-filling number" will be similarly a useful structural parameter.

6.2 Flood-Filling Games in Teaching Computational Thinking

From the beginning of time, humans have generated knowledge of the world around them and have developed procedures (or algorithms) in order to reach their goals [21]. This makes computer science crucial to the development of society, and modular thinking as a fundamental tool for human creative work [20].

In [22], an efficient and didactic method of teaching computational thinking (or algorithmic thinking) by introducing the concepts gradually is described. This is done by using the spiral curriculum, which goes back to the research of Jerome Bruner in the 1960s. The Center for Computer Science Education of ETH Zurich follows this principle when designing teaching materials. The increase in knowledge occurs gradually, along with the development of intuition and the ability to abstract. Thus, students can be introduced to computational thinking from primary school, learning to develop their programs in a modular and structured way.

The elaboration of the spiral curriculum has many levels of depth [21], making it an excellent tool for developing algorithmic thinking. The algorithm concept is very abstract and can be introduced gradually. Initially, one can work with specific inputs of a problem and intuitive strategies. Then, one can work with more robust strategies, analyzing the universe of viable solutions when possible. At a further level, one can analyze the quality of the solutions, and then look for good solutions, when the analysis of all solutions is not possible anymore.

One of the well-studied methods for teaching computational thinking is the use of digital games. We can find in the literature both studies of the didactic use of games already available and commonly used for recreation [4] and the development of games and platforms specifically aimed at computer education [22,23].

The flood-filling game is available on a variety of websites and applications and it can be configured for board dimensions and number of colors. Using the spiral approach, one can initially propose matches to the students with a small board and a small number of colors so that they find viable solutions and even list all of them. At this stage, they may observe that certain flooding moves do not change the number of tiles flooded, while other choices increase the flood. They may notice that among the colors that increase the flood, some increase it more than others. When comparing all the solutions found, they learn to measure the quality of each solution, noting that some have fewer moves than others. The goal of the game is then understood and the optimal solutions are those that have the least number of moves. Looking at these solutions, some concepts of simple and robust strategies, such as greedy, can be introduced. As the size of the board and the number of colors is increased, finding all the solutions is no longer a feasible task. Students are then prompted to search for heuristics that find a good solution. At this step, one can observe and discuss some algorithm development strategies, which can guarantee optimal solutions for some problems but not for others, such as flood-filling, and it is possible to explain the concept of NP-difficulty. The flood-filling game is useful for demonstrating and teaching many basic concepts of computation, from trivial definitions about graphs such as “neighborhood” and “the distance between two vertices”, to more complex concepts. Clearly, a study on which concepts should be inserted at each age and school subject is needed [20].

References

1. Arthur, D., Clifford, R., Jalsenius, M., Montanaro, A., Sach, B.: The complexity of flood filling games. In: Boldi, P., Gargano, L. (eds.) FUN 2010. LNCS, vol. 6099, pp. 307–318. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13122-6_30
2. Aschwanden, C.: Spatial simulation model for infectious viral disease with focus on sars and the common flu. In: 37th Annual Hawaii International Conference on System Sciences, HICSS (2004)
3. Barone, P., Bonizzoni, P., Vedova, G.D., Mauri, G.: An approximation algorithm for the shortest common supersequence problem: an experimental analysis. In: ACM Symposium on Applied, Computing, pp. 56–60 (2001)
4. Becker, K.: Teaching with games: the minesweeper and asteroids experience. J. Comput. Sci. Coll. **17**, 23–33 (2001)
5. Bodlaender, H.L., Fellows, M.R., Hallett, M.T., Wareham, T., Warnow, T.: The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs. Theor. Comput. Sci. **244**, 167–188 (2000)
6. Chor, B., Fellows, M., Ragan, M.A., Razgon, I., Rosamond, F., Snir, S.: Connected coloring completion for general graphs: algorithms and complexity. In: Lin, G. (ed.) COCOON 2007. LNCS, vol. 4598, pp. 75–85. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73545-8_10
7. Clifford, R., Jalsenius, M., Montanaro, A., Sach, B.: The complexity of flood-filling games. Theory Comput. Syst. **50**(1), 72–92 (2012)
8. dos Santos Souza, U., Rosamond, F., Fellows, M.R., Protti, F., da Silva, M.D.: The flood-it game parameterized by the vertex cover number. In: LAGOS 2015 - VIII Latin-American Algorithms, Graphs and Optimization Symposium, Electronic Notes in Discrete Mathematics, vol. 50, pp. 35–40 (2015)
9. Fellows, M.R., Fertin, G., Hermelin, D., Vialette, S.: Sharp tractability borderlines for finding connected motifs in vertex-colored graphs. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 340–351. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73420-8_31
10. Fellows, M.R., Hallett, M.T., Stege, U.: Analogs and duals of the mast problem for sequences and trees. J. Algorithms **49**(1), 192–216 (2003)
11. Fellows, M.R., Hallett, M.T., Wareham, H.T.: DNA physical mapping: three ways difficult. In: Lengauer, T. (ed.) ESA 1993. LNCS, vol. 726, pp. 157–168. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57273-2_52
12. Fellows, M., Protti, F., Rosamond, F., da Silva, M.D., Souza, U.S.: Algorithms, kernels and lower bounds for the flood-it game parameterized by the vertex cover number. Discrete Appl. Math. **245**, 94–100 (2017)
13. Fellows, M.R., dos Santos Souza, U., Protti, F., da Silva, M.D.: Tractability and hardness of flood-filling games on trees. Theor. Comput. Sci. **576**, 102–116 (2015)
14. Fleischer, R., Woeginger, G.J.: An algorithmic analysis of the honey-bee game. In: Boldi, P., Gargano, L. (eds.) FUN 2010. LNCS, vol. 6099, pp. 178–189. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13122-6_19
15. Fukui, H., Otachi, Y., Uehara, R., Uno, T., Uno, Y.: On complexity of flooding games on graphs with interval representations. In: Akiyama, J., Kano, M., Sakai, T. (eds.) TJJCCGG 2012. LNCS, vol. 8296, pp. 73–84. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45281-9_7
16. Golumbic, M., Kaplan, H., Shamir, R.: On the complexity of dna physical mapping. Adv. Appl. Math. **15**, 251–261 (1994)

17. Gusfield, D.: Efficient algorithms for inferring evolutionary tree. Networks **21**, 19–28 (1981)
18. Hallett, M.T.: An integrated complexity analysis of problems from computational biology. Ph.D. thesis, University of Victoria (1996)
19. Hon, W.-K., Kloks, T., Liu, F.-H., Liu, H.-H., Wang, H.-L.: Flood-it on at-free graphs. arXiv preprint arXiv:1511.01806 (2015)
20. Hromkovič, J.: Homo informaticus: why computer science fundamentals are an unavoidable part of human culture and how to teach them. Bull. EATCS **115**, 112–122 (2015)
21. Hromkovič, J., Lacher, R.: The Computer science way of thinking in human history and consequences for the design of computer science curricula. In: Dagiene, V., Hellas, A. (eds.) ISSEP 2017. LNCS, vol. 10696, pp. 3–11. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71483-7_1
22. Hromkovic, J., Kohn, T., Komm, D., Serafini, G.: Algorithmic thinking from the start. In: The Education Column, Bulletin of the EATCS, p. 121 (2017)
23. Hromkovič, J., Serafini, G., Staub, J.: XLogoOnline: a single-page, browser-based programming environment for schools aiming at reducing cognitive load on pupils. In: Dagiene, V., Hellas, A. (eds.) ISSEP 2017. LNCS, vol. 10696, pp. 219–231. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71483-7_18
24. Lacroix, V., Fernandes, C.G., Sagot, M.F.: Motif search in graphs: application to metabolic networks. IEEE/ACM Trans. Comput. Biol. Bioinf. **3**(4), 360–368 (2006)
25. Lagoutte, A., Noual, M., Thierry, E.: Flooding games on graphs. Discrete Appl. Math. **164**, 532–538 (2014)
26. Lampis, M.: Algorithmic meta-theorems for restrictions of treewidth. Algorithmica **64**(1), 19–37 (2012)
27. McMorris, F.R., Warnow, T.J., Wimer, T.: Triangulating vertex-colored graphs. SIAM J. Discrete Math. **7**(2), 296–306 (1994)
28. Meeks, K., Scott, A.: The complexity of flood-filling games on graphs. Discrete Appl. Math. **160**, 959–969 (2012)
29. Meeks, K., Scott, A.: The complexity of free-flood-it on $2 \times n$ boards. Theor. Comput. Sci. **500**, 25–43 (2013)
30. Meeks, K., Scott, A.: Spanning trees and the complexity of flood-filling games. Theory Comput. Syst. **54**(4), 731–753 (2014)
31. Meeks, K., Vu, D.K.: Extremal properties of flood-filling games. arXiv preprint arXiv:1504.00596 (2015)
32. Middendorf, M.: More on the complexity of common superstring and supersequence problems. Theor. Comput. Sci. **125**, 205–228 (1994)
33. Moran, S., Snir, S.: Convex recolorings of strings and trees: definitions, hardness results and algorithms. In: Dehne, F., López-Ortiz, A., Sack, J.-R. (eds.) WADS 2005. LNCS, vol. 3608, pp. 218–232. Springer, Heidelberg (2005). https://doi.org/10.1007/11534273_20
34. Seymour, P.D., Thomas, R.: Call routing and the ratcatcher. Combinatorica **14**, 217–241 (1994)
35. Woeginger, G.J., Fleischer, R.: An algorithmic analysis of the honey-bee game. Theor. Comput. Sci. **452**, 75–87 (2012)
36. Rahmann, S.: The shortest common supersequence problem in a microarray production setting. Bioinformatics **19**(Suppl. 2), ii156–ii161 (2003)
37. Raiha, K.-J., Ukkonen, E.: The shortest common supersequence problem over binary alphabet is NP-complete. Theor. Comput. Sci. **16**, 187–198 (1981)
38. Sim, J., Park, K.: The consensus string problem for a metric is NP-complete. J. Discrete Algorithms **1**(1), 111–117 (2003)

39. Souza, U.S., Protti, F., Dantas da Silva, M.: Inundação em grafos. In: Proceedings of the 16th Congreso Latino Iberoamericano de Investigación Operativa & 44th Simpósio Brasileiro de Pesquisa Operacional, CLAIO/SBPO 2012 (2012)
40. dos Santos Souza, U., Protti, F., da Silva, M.D.: Parameterized complexity of flood-filling games on trees. In: Du, D.-Z., Zhang, G. (eds.) COCOON 2013. LNCS, vol. 7936, pp. 531–542. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38768-5_47
41. dos Santos Souza, U., Protti, F., Silva, M.: An algorithmic analysis of flood-it and free-flood-it on graph powers. Discrete Math. Theor. Comput. Sci. **16**(3), 279 (2014)